Extended Abstract: The Categorical Structure of Semi-Bilinear Intuitionistic Logic

Harley Eades*
University of Iowa

1 Introduction

In [11] T. Crolard introduced Subtractive Logic (SL) a conservative extension of intuitionistic logic, where for every connective in the logic their dual is also a connective of the logic. In particular the logic is extended with the dual to implication called subtraction. Crolard also showed that the categoical interpretation of SL in bicartesian closed categories extended with coexponentials degenerates to a preorder. This shows that the chosen categorical interpretation does not capture the structure of the theory appropriately. In this talk I propose a categorical model of a logic called Semi-Bilinear Intuitionistic Logic (SBILL). This is the linearized version of SL. It is well known that the combination of connectives in SBILL have non-trivial models. This is the first linear subtractive logic complete with a categorical semantics.

The main point of this talk is the introduction of a new linear intuitionistic logic called Semi-Bilinear Logic (SBILL) (Section 4), and its categorical model (Section 5). However, before discussing these two main points I first motivate two key ideas. The first is the study of duality in computation, covered in Section 2, and the second is the use of category theory as a semantic framework, covered in Section 3.

2 Duality and Computation

Duality is easily seen in classical logic. For example, it is easy to show that conjunction is dual to disjunction using the de Morgan's dualities. In classical logic we can go one step further and define a dual to implication: $A \wedge \neg B \stackrel{\text{def}}{=} \neg(A \Rightarrow B)$.

The dual to implication called subtraction, difference, or exclusion was first explicitly studied by Cecylia Rauszer [22, 23]. She developed a logic called Heyting-Brouwer logic which used Heyting algebras extended with subtraction by taking the dual of implication. In computer science Crolard was the first to introduce the use of subtraction in SL. His main application is the study of coroutines in intuitionistic type theory [11, 12]. Crolard showed in [12] that the inhabitants of subtraction are coroutines which can only access their local environment.

Duality has been explored type theoretically and has revealed very interesting properties. Work exploiting duality in classical logics, with and without subtraction, showed that the call-by-value (CBV) evaluation strategy is dual to the call-by-name (CBN) evaluation strategy. This was shown first by Peter Selinger in [25] categorically, however, his logic did not have subtraction as a connective. The downside of Selinger's work is that it only proved the duality up to isomorphism. In response to Selinger's work Pierre-Louis Curien and Hugo Herbelin designed a classical type theory called the $\bar{\lambda}\mu\tilde{\mu}$-calculus which has subtraction as a connective. They proved the duality between CBV and CBN up to syntactic equality [13]. Philip Wadler did not care for subtraction and instead designed a type theory called the Dual Calculus which contained only negation, conjunction, and cut. He was then able to define λ-expressions, and arrow types using de Morgan's dualities. Wadler was also able to prove that CBV is dual to CBN [26, 27]

Dual to the notion of an inductive types are coinductive types which allow for the observation of infinite structures. It is possible to add coinduction to a terminating type theory while maintaining termination as long as one can show that the operations on infinite structures are always productive. A program is productive when, if it generates an infinite amount of data, then each piece of the infinite structure is generated in a finite amount of time. The study of type theories with both inductive and coinductive types is an important one. Many interesting structures can be defined using a mixture of inductive and coinductive operations [7]. However, dependent type theories with both inductive and coinductive types are not well understood. In fact coinductive types are not type safe in the core type theory of Coq [14], and in Agda it is impossible to mix inductive and coinductive definitions. I conjecture

*Advisor: Aaron Stump, Email: harley-eades@uiowa.edu

that logics like I discuss here will provide a framework where inductive and coinductive definitions can used in a type safe way, and be mixed. There are other recent projects trying to achieve the same goal [1, 2, 3]. However, none of these projects are approaching the problem from the direction of dualized logics.

3 Categorical Investigations

Terminating staticly-typed functional programming languages are isomorphic to logic. For example, the simply typed λ-calculus is isomorphic to intuitionistic propositional logic, while at the same time is a basic functional programming language. In the same vain the interactive theorem provers Coq and Agda can also be viewed as both higher order intuitionistic logics and dependently typed functional programming languages. This correspondence is part of a larger one, what I propose to call, the three perspectives of computation[1]. The three perspectives of computation is a correspondence between type theory, logic, and category theory. It asserts that all three are different takes on the same thing, computation, and interesting results in one will have interesting consequences in the others.

Language designers often want to extend their languages, and because we use the same language for logic as we do for programming we must be sure that any extensions do not result in any inconsistencies. So how do we guarantee our extensions are sound? This is the job of a well-developed semantics.

Establishing metaproperties, like consistency, of a theory is just one use of semantics. Another use is identifying defined objects with mathematical ones. This allows one to conclude facts about the defined object which correspond to facts about the semantic object. This allows one to understand the structures definable in the theory. Establishing such relationships is the key application of a category theoretic semantics.

Category theory is the abstract study of functions, and using it as a semantic framework has greatly influenced how we program in purely functional programming languages. These types of functional programming languages do not have any types of side effects. That is there is not state, hence, no assignment, and there are no primitive forms of IO.

Eugenio Moggi exploited the amazing correspondence between type theory and category theory, and showed how monads can be viewed as different forms of computation. Using his results we can implement side effects including state, IO, and even partiality. Monadic computation is a true testament to the power of categorical semantics.

Inductive and coinductive types have an initial algebra and final coalgebra semantics which are categorical objects. One key property of this semantics is that final coalgebras have the productivity property. This semantics have been used extensively in the study of theories containing such types. Another property this semantics reveals is the connection between induction and least fixpoints, and coinduction and greatest fixpoints [16, 17]. Thus, using these results we can justify the correctness of functions implementing various fold operations.

At this point it is easy to see that categorical semantics is very powerful, and can be used to study the structure of logic, type theory, and general purpose programming languages. However, before this power can be properly harnessed one must first construct a categorical model of the theory. This is the precise objective of my current work.

4 Semi-Bilinear Logic

Linear logic is a substructural logic with the enforcement that in any proof every hypothesis is used exactly once. It was Jean-Yves Girard who first proposed linear logic as a refinement of classical logic and intuitionistic logic [15]. The key aspect of Girard's genus is that he thought to combine the dualities of classical logic with the constructive structure of the latter. It is this eye towards duality that suggest that linear logic will be a great candidate for the study of subtraction. It is surprising that Girard did not include it originally, however, it was not until Joachim Lambek that subtraction was added to linear logic under the name of Bilinear Logic [19, 18]. However, Bilinear Logic is classical and we are interested in intuitionistic linear logic.

Semi-Bilinear Intuitionistic Logic (SBILL) consists of every connective of full intuitionistic linear logic, plus the dual of each of those connectives which includes subtraction. I call it "Semi-Bilinear" because it does not have all the metaproperties of bilinear intuitionistic logic. I adopt a simpler formalization, but sacrifice the cut-elimination property. This is not a terrible sacrifice, because cut elimination already fails for full intuitionistic linear logic [24]. The defintion of SBILL is in Section A. I denote the dual, linear subtraction, to linear implication, $A \multimap B$, as $A \bullet\!\!- B$. The judgment of SBILL is of the form of $\Gamma \vdash \Delta$ where Γ is the set of hypothesis, and Δ is the set of conclusions.

[1] This correspondence is also known as the Curry-Howard isomorphism or the propositions-as-types proofs-as-programs correspondence.

5 A Categorical Model

As I said before category theory is the abstract study of functions. Its most fundamental structure is the category.

Definition 1. *A **category** consists of the following data:*

- *a collection of objects which I denote by* $A, B, C, D, \ldots$,
- *a collection of morphisms (also called arrows) which I denote by* $f, g, h, j, \ldots$,
- *two assignments src and tar which assign a source object, and a target object to each morphism. For any morphism* f *if* $src(f) = A$ *and* $tar(f) = B$*, then I write* $f : A \to B$,
- *an identity morphism* $id_A : A \to A$,
- *for any morphisms* $f : A \to B$ *and* $g : B \to C$ *there must exist a morphism* $g \circ f : A \to C$ *called the composition of* f *and* g,
- *and the following equalities must hold:*
  - *for any morphism* $f : A \to B$, $id_B \circ f = f$, *and* $f \circ id_A = f$, *and*
  - *for any morphisms* $f : A \to B$, $g : B \to C$, *and* $h : C \to D$, $(h \circ g) \circ f = h \circ (g \circ f)$.

A logic is interpreted in a categorical model by finding a category suitable for interpreting formulas as objects of the category and sequents as morphisms [20]. This means there must exist a suitable interpretation of contexts. Assuming one exists, then $[\![\Gamma \vdash \Delta]\!] = \phi : [\![\Gamma]\!] \to [\![\Delta]\!]$ where ϕ is a morphism in the desired category. However, before such an interpretation can be defined, we must first find a category with the structure which captures ones logic.

The categorical model of intuitionistic linear logic is well understood. A complete model can be found in the work of Gavin Bierman [6, 5] called Linear Categories. These are essentially symmetric monoidal categories. That is the category is extended with the notion of tensor with some suitable equalities. Monoidal categories go all the way back to Saunders Mac Lane, one of the founders of category theory. Full intuitionistic linear logic contains only half of SBILL. Its connectives are linear implication, tensor, conjunction and disjunction, their units, and of-course (denoted !). Now cointuitionistic linear logic has the dual of all the previous connectives. Thus, it contains linear subtraction, par, conjunction, disjunction, their units, and the dual to of-course called why-not (denoted ?). Gianluigi Bellin showed in [4] that by taking the dual of a Bierman's linear categories one obtains a categorical model of co-intuitionistic linear logic. Thus, to obtain a categorical model of SBILL one must take the union of these two models. However, this union does not capture all of the structure of SBILL. It will not capture the fact that SBILL has multiple conclusions.

To model multiple conclusions additional axioms connecting the two sides of the model are needed. These axioms place a strength constraint on one of the two connectives. They enforce that tensor is stronger than par. This then allows for the interpretation of hypothesis of a sequent to be equal to the tensor of all the formulas, and the interpretation of the conclusions to be equal to the par of all the formulas. These axioms can be imported into our model by requiring the model to be linearly distributive. Linearly distributive categories have been rigorously studied by J.R.B. Cockett, R.A.G. Seely, R.F. Blute, and T.H. Trimble. I cite only a few papers in this line of work [8, 9, 10]. Using these facts we can now prove that a category that is a linear category, a co-linear category, and is linearly distributive is a model of SBILL:

Theorem 2. *A categorical model of SBILL is any category that is a linear category, is a co-linear category, and is linearly distributive.*

6 Conclusion

I motivated the study of duality and computation by discussing some of the key points in the literature, and then motivated the use of category theory as a semantic framework. Finally, I introduced Semi-Bilinear Logic (SBILL), and its categorical model.

Future Work. The main property I wish to prove in the future is that adding inductive types and adding the obvious dual of inductive types will yield coinductive types with the productivity property without any syntactic restrictions. In addition I hope to show that these can be freely mixed. To prove these properties the categorical model introduced here will be invaluable.

Linear logic has been shown to have a resource semantics. What this means is that one can interpret each formula as a resource. The linear property on formulas now enforces that each resource can be used only once. Now using this style of semantics one can prove properties of concurrent programs [21]. The question I would like to investigate is "does the presence of perfect duality yield any new and interesting properties?" This has never been explored.

References

[1] A. Abel. Wellfounded recursion with copatterns: A unified approach to termination and productivity. *Accepted for presentation at ICFP 2013*, 2013.

[2] A. Abel, B. Pientka, D. Thibodeau, and A. Setzer. Copatterns – programming infinite structures by observations. *Proceedings of POPL'13*, January 2013.

[3] R. Atkey and C. McBride. Productive coprogramming with guarded recursion. In *18th ACM SIGPLAN International Conference on Functional Programming (ICFP 2013)*, 2013. To appear.

[4] G. Bellin. Categorical proof theory of co-intuitionistic linear logic. *LOMECS*, 2012.

[5] G. Bierman. What is a categorical model of intuitionistic linear logic? In M. Dezani-Ciancaglini and G. Plotkin, editors, *Typed Lambda Calculi and Applications*, volume 902 of *Lecture Notes in Computer Science*, pages 78–93. Springer Berlin Heidelberg, 1995.

[6] G. M. Bierman. *On Intuitionistic Linear Logic*. PhD thesis, Wolfson College, Cambridge, December 1993.

[7] V. Capretta. Wander types : A formalization of coinduction-recursion. *Progress in Informatics*, (10):47–64, 2013.

[8] J. Cockett and R. Seely. Proof theory for full intuitionistic linear logc, bilinear logic, and mix categories. *Theory and Applications of Categories*, 3(5):85–131, 1997.

[9] J. Cockett and R. Seely. Weakly distributive categories. *Journal of Pure and Applied Algebra*, 114(2):133 – 173, 1997.

[10] J. Cockett and R. Seely. Linearly distributive functors. *Journal of Pure and Applied Algebra*, 143(1–3):155 – 203, 1999.

[11] T. Crolard. Subtractive logic. *Theor. Comput. Sci.*, 254(1-2):151–185, 2001.

[12] T. Crolard. A formulae-as-types interpretation of subtractive logic. *J. Log. and Comput.*, 14(4):529–570, Aug. 2004.

[13] P.-L. Curien and H. Herbelin. The duality of computation. In *Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, ICFP '00, pages 233–243, New York, NY, USA, 2000. ACM.

[14] C. E. Giménez. *Un Calcul De Constructions Infinies Et Son Application A La Verification De Systemes Communicants*. PhD thesis, Ecole Normale Superieure de Lyon, 1997.

[15] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1 – 101, 1987.

[16] A. D. Gordon. A tutorial on co-induction and functional programming. 1995.

[17] B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62:62–222, 1997.

[18] J. Lambek. *Substructural Logics*, volume 2, chapter From Categorical Grammar to Bilinear Logic. Oxford Science Publications, 1993.

[19] J. Lambek. Cut elimination for classical bilinear logic. *Fundam. Inform.*, 22(1/2):53–67, 1995.

[20] J. Lambek and P. Scott. *Introduction to Higher-Order Categorical Logic*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1988.

[21] K. Mazurak and S. Zdancewic. Lolliproc: to concurrency from classical linear logic via curry-howard and control. *SIGPLAN Not.*, 45(9):39–50, Sept. 2010.

[22] C. Rauszer. Semi-boolean algebras and their applications to intuitionistic logic with dual operations,. *Fundamenta Mathematicae*, 83:219–249, 1974.

[23] C. Rauszer. Applications of kripke models to heyting-brouwer logic. *Studia Logica*, 36:61–71, 1977.

[24] H. SCHELLINX. Some syntactical observations on linear logic. *Journal of Logic and Computation*, 1(4):537–559, 1991.

[25] P. Selinger. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(02):207–260, 2001.

[26] P. Wadler. Call-by-value is dual to call-by-name. *SIGPLAN Not.*, 38(9):189–201, Aug. 2003.

[27] P. Wadler. Call-by-value is dual to call-by-name – reloaded. In J. Giesl, editor, *Term Rewriting and Applications*, volume 3467 of *Lecture Notes in Computer Science*, pages 185–203. Springer Berlin Heidelberg, 2005.

A Definition of SBILL

Syntax:

$$\begin{array}{llll} \text{(Formulas)} & A, B, C & ::= & \bot \mid \top \mid 1 \mid 0 \mid A \multimap B \mid A \bullet\!\!- B \\ & & & \mid A \otimes B \mid A \oplus B \mid A \times B \\ & & & \mid A + B \mid !A \mid ?A \\ \text{(Contexts)} & \Gamma, \Delta & ::= & \cdot \mid A \mid \Gamma, \Gamma' \end{array}$$

Inference Rules:

$$\frac{}{A \vdash A}\ \textsc{ax} \qquad \frac{}{\cdot \vdash 1}\ \textsc{tunit}\mathrm{R}$$

$$\frac{}{0 \vdash \cdot}\ \textsc{punit}\mathrm{L} \qquad \frac{}{\Gamma \vdash \top, \Delta}\ \textsc{cunit}$$

$$\frac{}{\Gamma, \bot \vdash \Delta}\ \textsc{dunit} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, 1 \vdash \Delta}\ \textsc{tunit}\mathrm{L}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash 0, \Delta}\ \textsc{punit}\mathrm{R} \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ \textsc{cut}$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \multimap B \vdash \Delta, \Delta'}\ \textsc{imp}\mathrm{L} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B, \Delta}\ \textsc{imp}\mathrm{R}$$

$$\frac{A \vdash B, \Delta}{\Gamma, A \bullet\!\!- B \vdash \Delta}\ \textsc{sub}\mathrm{L} \qquad \frac{\Gamma' \vdash A, \Delta' \quad \Gamma, B \vdash \Delta}{\Gamma, \Gamma' \vdash A \bullet\!\!- B, \Delta, \Delta'}\ \textsc{sub}\mathrm{R}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ \textsc{tensor}\mathrm{L} \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \otimes B, \Delta, \Delta'}\ \textsc{tensor}\mathrm{R}$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \oplus B \vdash \Delta, \Delta'}\ \textsc{par}\mathrm{L} \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \oplus B, \Delta}\ \textsc{par}\mathrm{R}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \times B \vdash \Delta}\ \textsc{conjl}\mathrm{L} \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \times B \vdash \Delta}\ \textsc{conjl}\mathrm{R}$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \times B, \Delta}\ \textsc{conj}\mathrm{R} \qquad \frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A + B \vdash \Delta}\ \textsc{disj}\mathrm{L}$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A + B, \Delta}\ \textsc{disjr}\mathrm{L} \qquad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A + B, \Delta}\ \textsc{disjr}\mathrm{R}$$

$$\frac{\Gamma, B, A, \Gamma' \vdash \Delta}{\Gamma, A, B, \Gamma' \vdash \Delta}\ \textsc{ex}\mathrm{L} \qquad \frac{\Gamma \vdash \Delta, B, A, \Delta'}{\Gamma \vdash \Delta, A, B, \Delta'}\ \textsc{ex}\mathrm{R}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta}\ \textsc{oc}\mathrm{L} \qquad \frac{!\Gamma \vdash A, ?\Delta}{!\Gamma \vdash !A, ?\Delta}\ \textsc{oc}\mathrm{R}$$

$$\frac{\Gamma, !A, !A, \Gamma' \vdash \Delta}{\Gamma, !A, \Gamma' \vdash \Delta}\ \textsc{oc}\mathrm{C} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta}\ \textsc{oc}\mathrm{W}$$

$$\frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta}\ \textsc{wn}\mathrm{L} \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash ?A, \Delta}\ \textsc{wn}\mathrm{R}$$

$$\frac{\Gamma \vdash ?A, ?A, \Delta}{\Gamma \vdash ?A, \Delta}\ \text{wNC} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash ?A, \Delta}\ \text{wNW}$$